OOP teniszklub dokumentáció

Feladat:

Modellezzük egy teniszklub működését!

A klub nyilvántartja a klubtagjait, akik foglalást tehetnek a klub szabad teniszpályáira. A teniszpályák borítása lehet fű, salak, vagy műanyag, néhányuk fedett. A pályákat sorszám azonosítja. Minden foglalás 1 órára szól, amely tartalmazza a foglaló klubtag nevét, a választott pálya sorszámát, a foglalás dátumát, és a lefoglalt órát (6-20 közötti szám). A füves pálya óradíja 5000 Ft, a salakosé 3000 Ft, a műanyagé 2000 Ft, de ezt módosíthatja egy szorzó attól függően, hogy a klubtag igazolt sportoló, diák vagy nyugdíjas-e, továbbá az így kalkulált díjra 20% felár is kerül, ha a pálya fedett.

Tegye lehetővé, hogy a klub új pályát tudjon létesíteni, egy régit fel tudjon számolni; egy tagot be-, illetve ki tudjon léptetni, egy tag időpontot tudjon foglalni egy pályára, amit akár vissza is mondhat a foglalási időpont előtt.

Meg lehessen válaszolni az alábbi kérdéseket:

a. Keressünk egy adott időpontra megadott borítású szabad pályákat.

b. Mondjuk meg mely pályákat foglalta le egy tag egy adott napra és mely órákra?

c. Mennyi pályahasználati díjat kell fizetnie az adott napra egy adott tagnak?

d. Mennyi a teniszklub bevétele egy adott időszakra (kezdő és vég dátum között)?

Készítsen használati eset diagramot a klub és egy klubtag szempontjából! Ebben jelenjenek meg használati esetként a később bevezetett fontosabb metódusok. Adjon meg a fenti feladathoz egy olyan objektum diagramot, amely mutat öt pályát, két teniszklubtagot, ezekhez kapcsolódó 2-2 pályafoglalást. Egy kommunikációs diagramban jelölje, hogy mely objektumok milyen metódusokkal kell, hogy rendelkezzenek ahhoz, hogy a kívánt funkcionalitást biztosítani tudjuk.

Készítse el egy teniszpálya objektum állapotgépét! Különböztesse meg a „nincs foglalás”, és a „foglalások vannak” állapotokat. Az állapot-átmeneteket megvalósító tevékenységeket majd a pálya osztály metódusaiként definiálhatja. Egészítse ezt ki kommunikációs diagrammá!

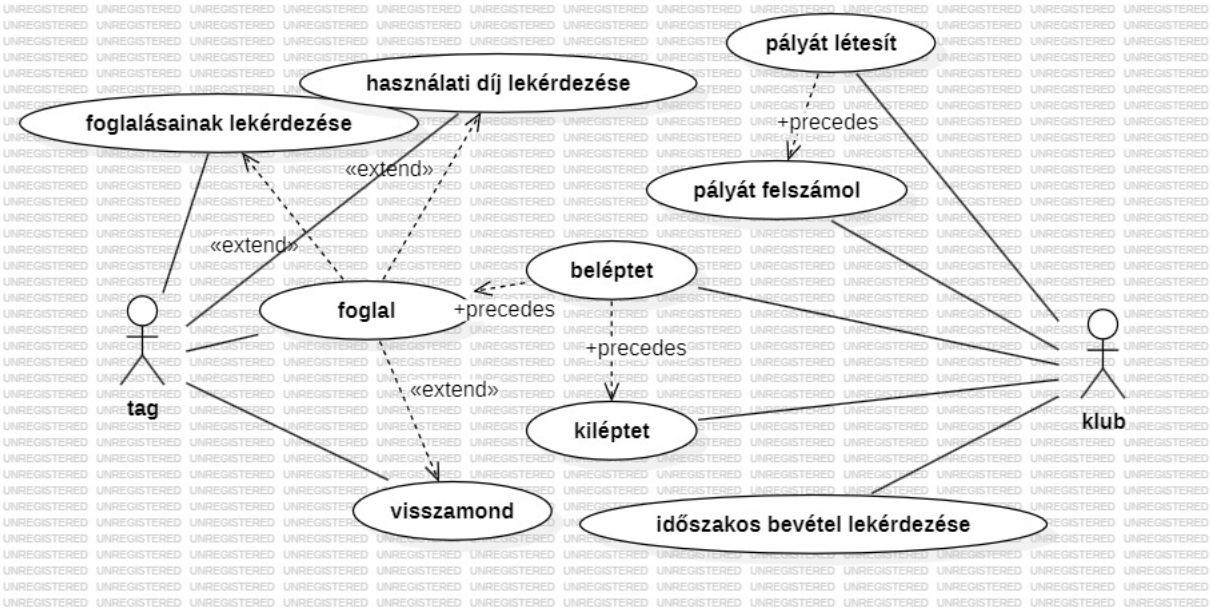
Rajzolja fel a feladat osztály diagramját! Felteheti, hogy a rejtett adattagokhoz mindig tartozik egy publikus getter: ha mégsem, akkor azt a „secret” megjegyzéssel jelölje. Egészítse ki az osztálydiagramot az objektum-kapcsolatokat létrehozó metódusokkal, valamint a feladat kérdéseit megválaszoló metódusokkal. A metódusok leírása legyen

minél tömörebb (például ciklusok helyett a megfelelő algoritmus minta specifikációs jelölését használja). Használjon tervezési mintákat, és mutasson rá, hogy hol melyiket alkalmazta.

Implementálja a modellt! Szerkesszen olyan szöveges állományt, amelyből fel lehet populálni egy teniszklub pályáit, tagjait, néhány foglalást is visszamondást. Válaszoljuk meg a b. c. d. kérdéseket. Készítsen teszteseteket, néhánynak rajzolja fel a szekvencia diagramját, és hozzon létre ezek kipróbálására automatikusan tesztkörnyezetet!

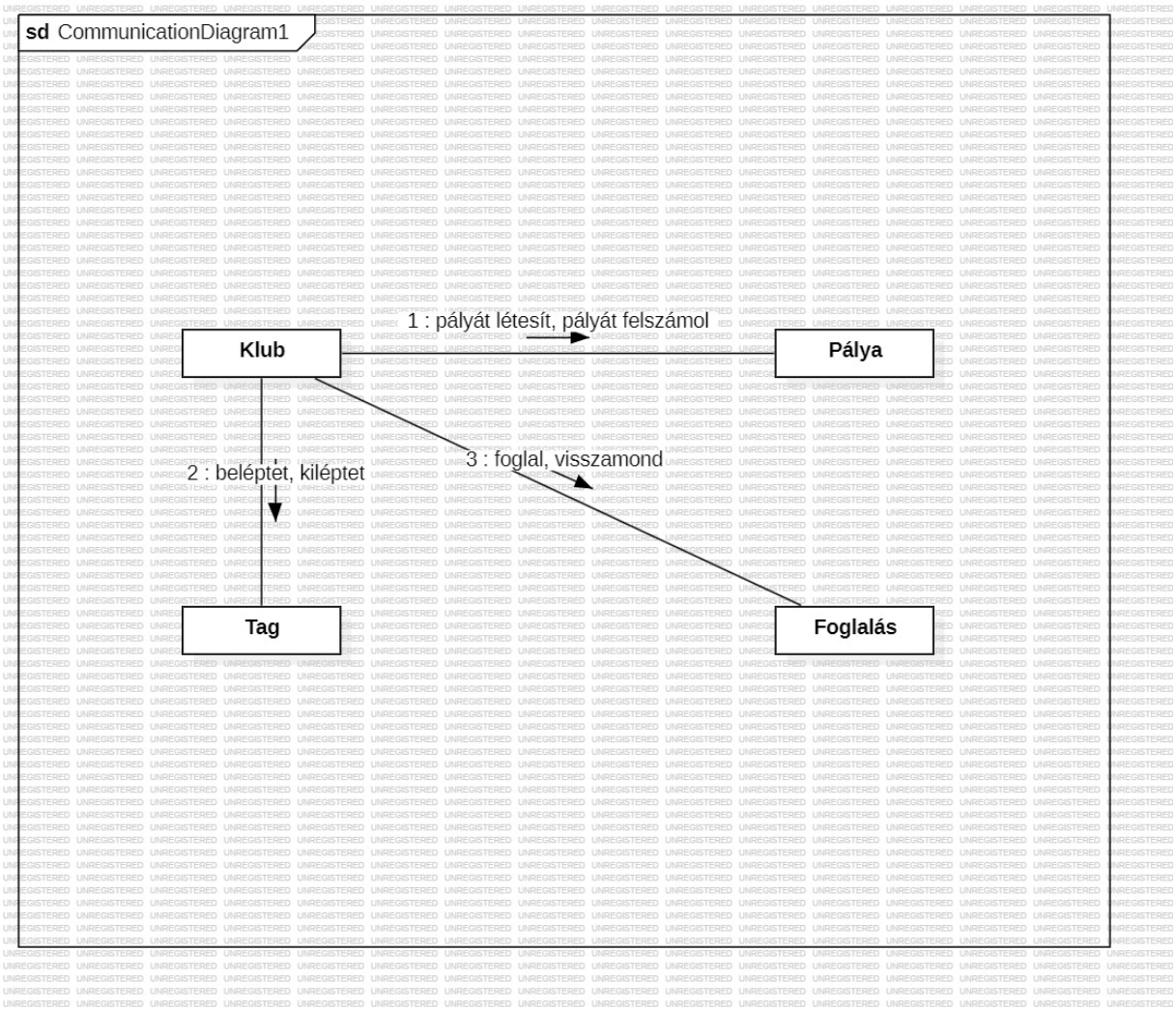
Használati eset diagram:

A klubtag fő tevékenységei a foglalás és esetlegesen a visszamondás, emellett egyéb hasznos tevékenységeket is hozzáadtam, mint például a saját foglalásainak a lekérdezését és a használati díj lekérdezését, mivel ezek fontosak lehetnek egy klubtag számára. A klub fontos tevékenységei közé tartozik a pálya létesítése, felszámolása, a tag beléptetése és kiléptetése, egy hasznos metódus emellett pedig még például az időszakos bevétel lekérdezése.



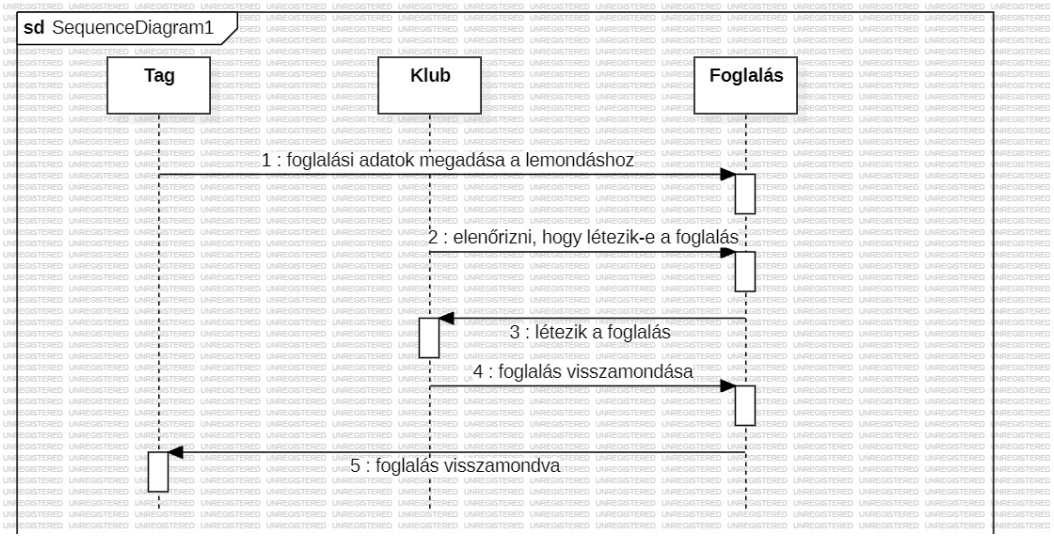
Kommunikációs diagram:

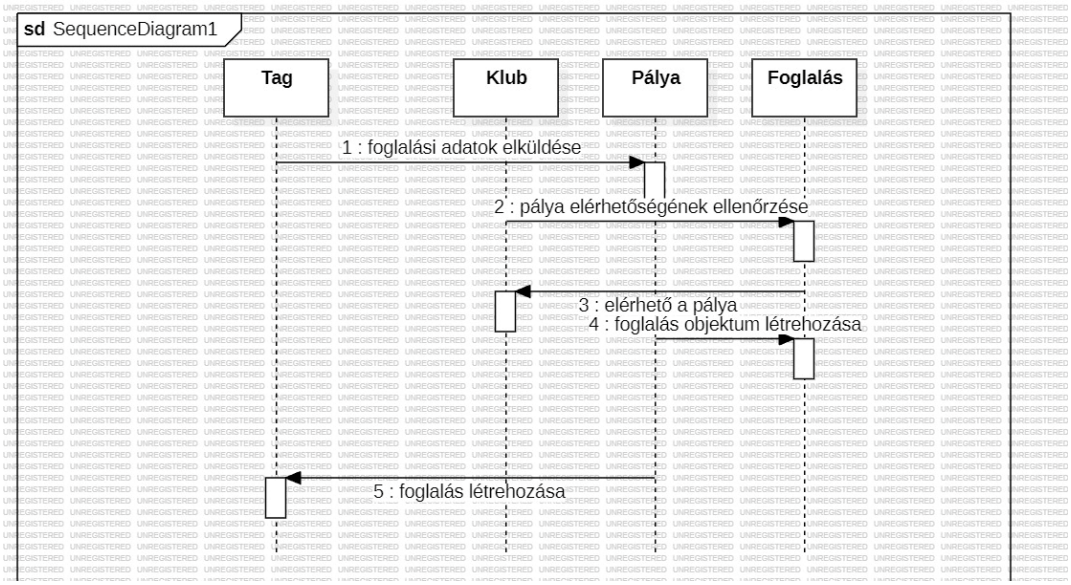
A pálya létesítése során a klub és pálya között, a tag belépése és kilépése során a klub és a tag között, a foglalás és visszamondás során pedig a klub és a foglalás osztályok között történik a kommunikáció, mivel a klub végzi a foglalás adminisztrálását.



Szekvencia diagramok:

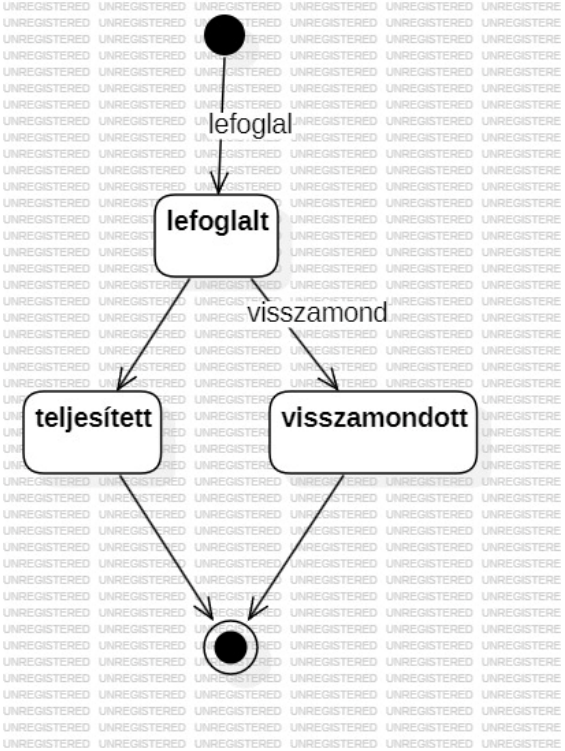
Az első egy foglalás visszamondását, a második pedig egy foglalás létrehozását modellezi.





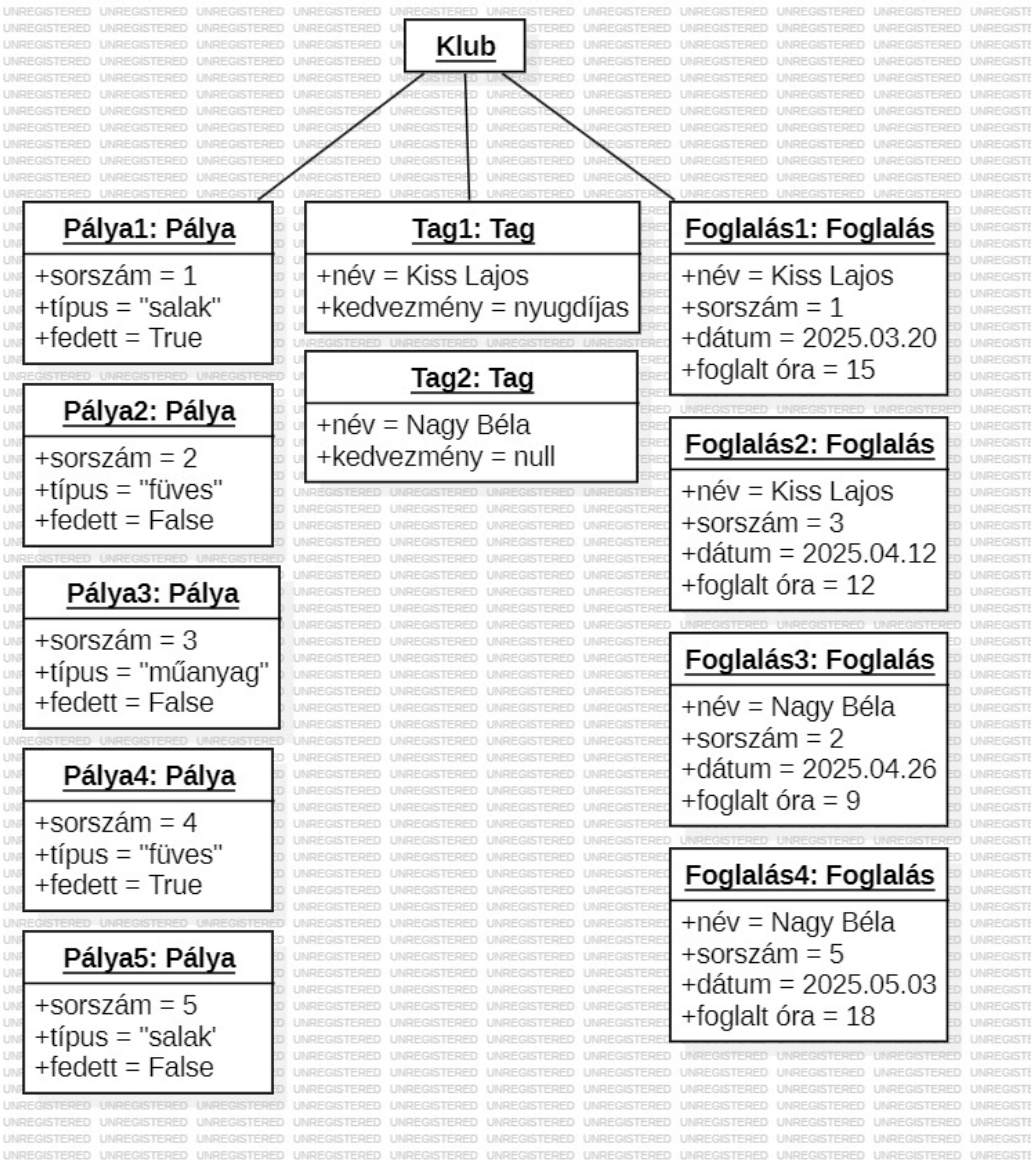
Állapotgép:

Foglalás után kétfajta kimenet lehet, teljesül a foglalás, vagy a klubtag visszamondja azt.



Objektumdiagram:

A klub osztály tartalmazza a Pálya, Tag és Foglalás osztályokat. A példán 5 különböző pálya, 2 klubtag és 4 foglalás látható. Van olyan pálya, amit senki nem foglalt le.

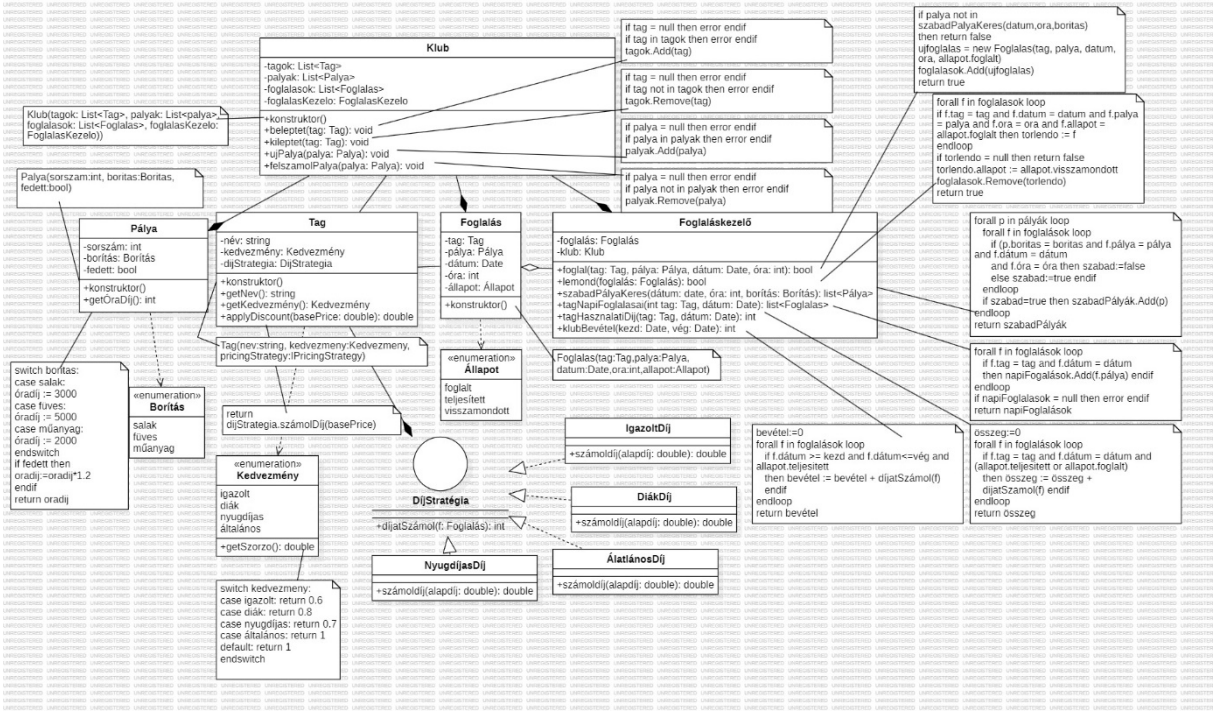


Osztálydiagram:

Klub, Tag, Foglaláskezelő és Pálya osztályok szerepelnek rajta, illetve a díjszámításhoz stratégia tervezési mintát alkalmaztam, mivel hasonlóan kell mindegyiket kiszámolni, viszont az adott tag kedvezményétől függ a szorzó, amivel a pálya alapdíját meg kell szorozni.

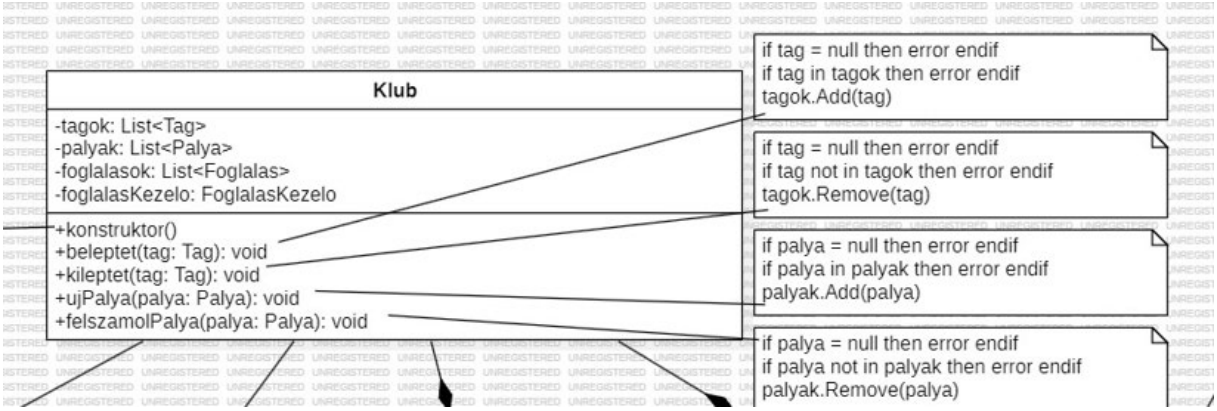
Statégia tervezési minta: Lehetővé teszi, hogy egy algoritmus vagy művelet cserélhető legyen futási időben anélkül, hogy az azt használó ügyfélkódot módosítani kellene. Elkerülhető vele a kódredundancia, és a SOLID elvek közül az open-closed elvnek eleget tesz, mivel módosításnál vagy újabb kedvezmény hozzáadásánál nem kell egy egész kódot átírni, csak egy újabb az interfészből leszármaztatott osztályt hozzáadni.

Teljes osztálydiagram:



Klub osztály:

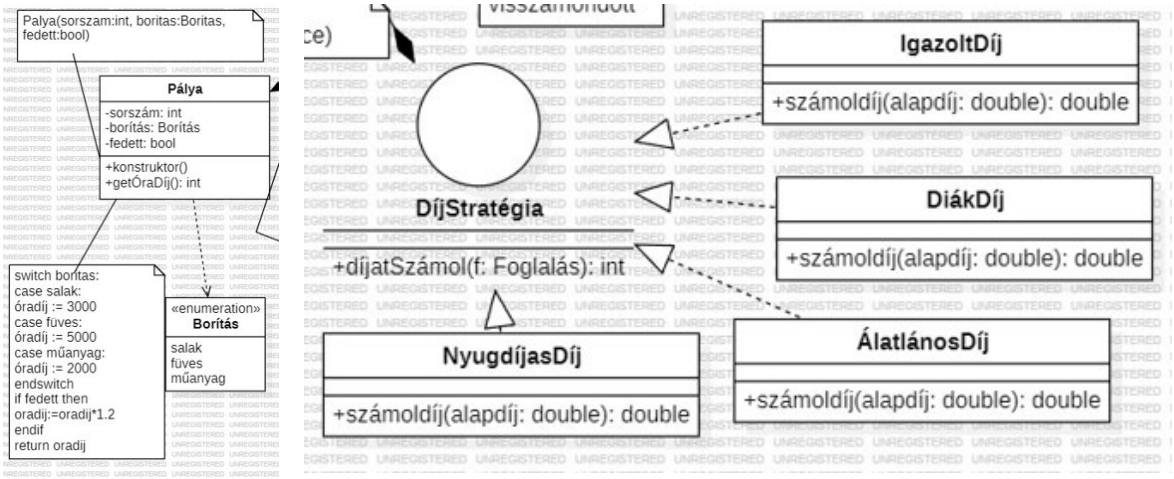
A klub tartalmazza a tagok, pályák, és foglalások listáját, illetve egy Foglaláskezelő példányt. A tag beléptetése, kiléptetése, és a pálya létrehozása, felszámolása metódusokban ellenőrzésre kerül, hogy az adott tag/pálya benne van-e már a listában, és ennek megfelelően kerül hozzáadásra vagy eltávolításra.



Pálya osztály és DíjStratégia:

A pálya osztály tartalmaz egy gettert az óradíj kiszámításához, ami az óradíj és a pálya fedettsége alapján számol. A tag osztály tartalmaz egy DíjStratégia példányt, illetve hozzá van kötve a DíjStratégia interfész is. A kedvezmény enumnak van egy getSzorzo

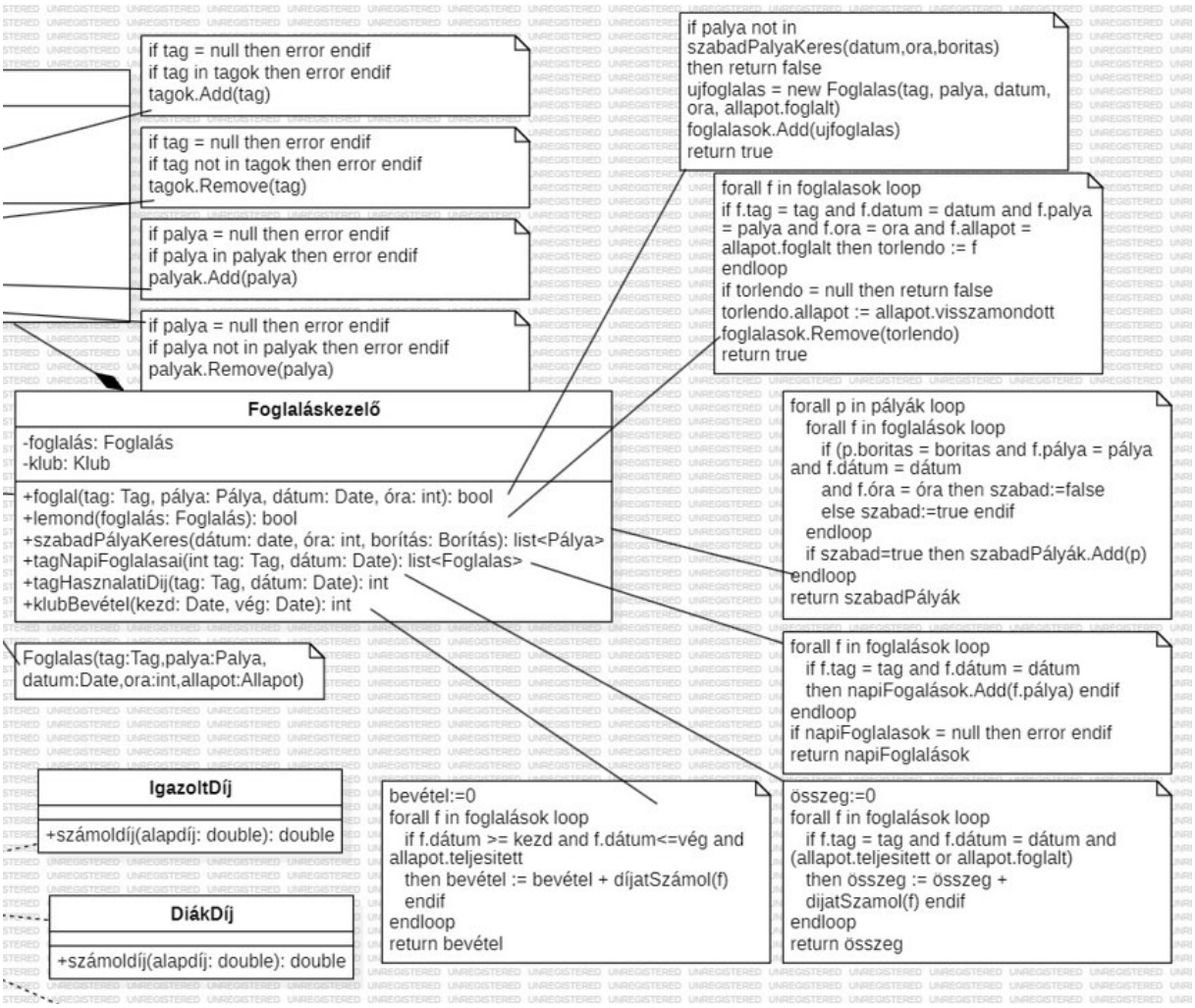
metódusa, ami a kedvezmény szorzóját adja vissza. A DíjStratégiában majd ez a szorzó és az alapár segítségével számoljuk ki a végső árat.



Foglaláskezelő osztály:

A foglal metódusnál meghívjuk a SzabadPályaKeres metódust, és ha az ezáltal visszaadott listában benne van az adott pálya, akkor azt le tudjuk foglalni, ezután létrehozunk egy új Foglalás objektumot, majd hozzáadjuk azt a foglalások listájához.
A visszamondásnál ellenőrizzük, hogy a foglalások listájában valóban szerepel-e a keresett foglalás, ha igen akkor az állapotát visszamondottra állítjuk, és töröljük a foglalások listájából.
A szabadPályaKeres metódusban az adott pályát a foglalások listában keressük, és ha ott nem találjuk, az azt jelenti hogy szabad. A végén a szabad pályákról egy listát adunk vissza.
A tagNapiFoglalásai megkeresi az összes olyan foglalást, ahol a tag neve egyezik és a megadott dátumon van, majd ezekről egy listát ad vissza.
A tagHasználatiDíj megkeresi a foglalások listában azokat a foglalásokat, ami a megadott taghoz tartozik és a megadott dátumon történt, majd ellenőrzi is az állapotát, így a visszamondottakat semmiképp nem számolja bele. Erre valójában nem is feltétlen van szükség, mivel a lemond metódusban az állapot „lemondott”-ra állítása mellett a foglalások listából is töröljük az adott foglalást. A feltételeknek megfelelő foglalásokhoz díjat is számol, majd hozzáadja az összeghez.
A klubBevétel metódus megkeresi azokat a foglalásokat, amik két megadott időpont között vannak, és az állapotuk teljesített. Interpretációtól függően itt akár a foglalt állapotúakat is bele lehetne számolni, viszont ha feltételezzük, hogy azokat még lemondhatják, akkor a bevétel még változhat. Ha a teljesített és foglalt állapotú foglalásokat is beleszámoljuk, akkor a tagHasználatiDíj metódushoz hasonlóan itt sem

feltétlen szükséges az állapotot ellenőrizni, mivel a visszamondottak törölve lettek a foglalások listából.



Tesztelési terv:

1. Tagkezelés tesztelése

1.1 Tag létrehozása és adatok ellenőrzése

- Teszteset: TagCreation_WithValidData_Success
- Leírás: Ellenőrzi, hogy a tag létrehozása megfelelően történik-e.
- Várt eredmény: A tag neve és kedvezménye helyesen állítható be.

1.2 Kedvezményes díjszámítás (Diák, Általános stb.)

- Teszteset: CalculateUsageFee_CorrectForDiak
- Leírás: Ellenőrzi, hogy a diák kedvezményt helyesen alkalmazzák-e a pályahasználati díjon.

- Várt eredmény: A diák kedvezmény (20%) helyesen csökkenti az óradíjat.

2. Pályák kezelése - Pályaóradíj számítása

- Teszteset: CourtPriceCalculation_FedettSalak_CorrectPrice
- Leírás: Ellenőrzi, hogy a fedett salak pálya óradíja helyesen számolódik-e (alapár × 1,2).
- Várt eredmény: 3600 Ft (3000 × 1,2).

3. Foglalási logika tesztelése

3.1 Sikeres foglalás

- Teszteset: Reservation_AvailableCourt_Success
- Leírás: Ellenőrzi, hogy egy szabad pályára történő foglalás sikeres-e.
- Várt eredmény: A foglalás hozzáadódik a listához.

3.2 Ütköző foglalás elutasítása

- Teszteset: Reservation_SameTimeSameCourt_Fails
- Leírás: Ellenőrzi, hogy egy már foglalt időpontra nem lehet új foglalást létrehozni.
- Várt eredmény: A második foglalás sikertelen.

3.3 Foglalás lemondása

- Teszteset: CancelReservation_Success
- Leírás: Ellenőrzi, hogy egy létező foglalás lemondható-e.
- Várt eredmény: A foglalás törlődik a listából.

4. Tag-specifikus funkciók tesztelése

4.1 Napi foglalások listázása

- Teszteset: TagNapiFoglalasai_ReturnsCorrectReservations
- Leírás: Ellenőrzi, hogy egy tag napi foglalásai helyesen jelennek-e meg.
- Várt eredmény: A visszaadott lista pontosan tartalmazza a megfelelő foglalásokat.